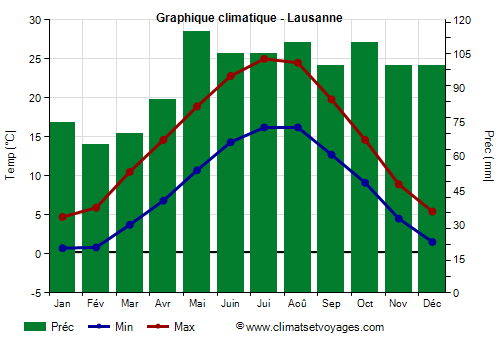
P\_FUN

Concevoir un logiciel pour afficher des graphiques sur des données



Matias Denis

ETML – FID2

Vennes – Lausanne

Table des matières

[Introduction du projet 3](#_Toc210048739)

[Objectifs du jeu 3](#_Toc210048740)

[Domaine d’application 3](#_Toc210048741)

[Planification 3](#_Toc210048742)

[Rapport de tests 4](#_Toc210048743)

[Journal de travail 4](#_Toc210048744)

[Usage de l’IA 4](#_Toc210048745)

[Conclusion 4](#_Toc210048746)

# Introduction du projet

Ce projet a pour but de réaliser un logiciel pour afficher des graphiques sur des données.

## Objectifs du jeu

Les objectifs du jeu sont :

* Réaliser un programme informatique de qualité.
  + Organisé
  + Compacté
  + Optimisé
  + Testé
  + Commenté
  + Complet
* Prouver que l’on est digne de confiance.
  + Journal de travail à jour
  + Pro-activité
    - Poser des questions au client
    - Faire des démonstrations
    - Utiliser un système de versioning de code (GIT)

## Objectifs pédagogiques

## Domaine d’application

Le domaine d’application dans lequel le logiciel est utilisé est la météo en Suisse. Plus précisément, les précipitations mesurées dans quatre villes suisses : Lausanne, Davos, Zürich et Lugano depuis environ 150 ans.

Ces données ont été récupérées sur le site officiel de météorologie et de climatologie MétéoSuisse.

# Planification

La planification a été faite sur GitHub Projects sous forme de monthly roadmap.

* 25 août 2025 -> Prise en main du projet
* 1 septembre 2025 -> Création de la maquette de l’interface graphique du produit.
* 8 septembre 2025 -> Début du code et donc de l’importation des fichiers CSV.
* 15 septembre 2025 -> Affichage d’une courbe simple.
* 29 septembre 2025 -> Affichage de plusieurs courbes, séries de données importées.
* 6 octobre 2025 -> Implémentation de la navigation et du zoom.
* 27 octobre 2025 -> Implémentation d’un mode de fonctions mathématiques.

# Production

Le projet est construit en C# avec Windows Forms pour l’interface graphique et ScottPlot v5 pour l’affichage des graphiques.

L’application permet de :

* Importer des données météorologiques (précipitations) depuis un fichier CSV (format important)
* Visualiser ces données sous formes de courbes
* Filtrer par ville et par période
* Appliquer éventuellement une fonction mathématique de démonstration

## Organisation du code

**MainWindow.xaml.cs** : contient la logique de l’interface. On y trouve les gestionnaires d’événements (boutons, cases à cocher/décocher, sélection de dates) et la méthode **Redraw()** qui met à jour le graphique.

**MeteoSuisseCsvImporter.cs :** classe utilitaire qui lit les fichiers CSV, détecte le séparateur, gère l’encodage, et transforme chaque ligne en objet **TimePoint**.

**TimePoint**(record) :représente un couple **(Date, Valeur)** correspondant à une mesure de précipitations.

**Extensions LINQ**: ajoutées pour faciliter le filtrage des données par période, obtenir les bornes, etc…

**FunctionFactory**: permet d’ajouter des fonctions mathématiques à tracer (utilisé comme démonstration) de fonctionnalités de ScottPlot).

## Choix techniques

ScottPlot v5 a été choisi pour la simplicité de génération de graphiques interactifs.

Le code utilise LINQ au lieu de boucles for, ce qui rend le traitement des séries plus lisible et concis.

La lecture du CSV a été rendue robuste : détection automatique du séparateur ( ; ou , ou tabulation), prise en compte de différents formats de date (ex : dd.mm.yyyym yyyy-mm-dd), gestion des décimales avec virgules ou points.

Une attention particulière a été portée à l’encodage (UTF-8 / Latin-1), car les fichiers contiennent des accents qui causent des erreurs si mal décodés.

## Problèmes rencontrés et solutions

**Encodage**: les fichiers de MétéoSuisse étaient en Latin-1, ce qui empêchait la reconnaissance correcte des colonnes. La solution a été de forcer la lecture avec **Encoding.Latin1**.

**Variation des en-tête CSV**: certains fichiers n’avaient pas exactement les mêmes intitulés de colonnes. La solution a été de normaliser les nom (minuscule, sans accents).

## Qualité et bonnes pratiques

Les méthodes sont courtes et bien séparées (import, affichage, extensions LINQ)

Le code est largement commenté et respecte les conventions de nommage de l’ETML (identifiants explicites, variables en camelCase, classes en PascalCase).

Les exceptions sont gérées proprement dans l’import (erreurs de format, dates invalides, séparateur inconnu). L’utilisateur reçoit un message clair dans la zone de statut.

# Rapport de tests

# Journal de travail

Le journal de travail retraçant toutes les étapes des différentes séquences est disponible dans le GitHub du projet. Le lien pour s’y rendre est ici https://github.com/Jazztel20/P\_FUN/blob/main/JDT/T-P\_FUN-MatiasDenis-JDT.xlsx

# Usage de l’IA

L’IA a servi de guide dans ce projet car les bases, jusqu’ici acquises en C#, ne permettent pas de produire un projet comme celui-ci sans aide externe. De plus, tous les projets antérieurs ont été effectués en console et non en Windows Forms, ce qui rajoute de la difficulté. Il a été demandé à l’IA de produire ce qui prenait le plus de temps, le code et le debug des erreurs.

# Conclusion

Ce projet est très difficile à faire pour une personne n’ayant pas le niveau requis en C# et en Windows Forms. Beaucoup de fonctionnalités comme l’import de CSV, l’affichage d’une fenêtre graphique où les données des fichiers CSV sont affichées sont des concepts nullement vus en cours. Le code basique et, lui aussi, dur à produire car les bases de C# ne sont pas acquises de mon côté. Je trouve ce projet peu pertinent aux vues du niveau global en programmation C# & Windows Forms. Je n’y ai trouvé que très peu d’intérêt.